club commodore

Boletín informativo para los usuarios de microordenadores VIC y CBM



Nº/3

diciembre 1982



UN SALUDO DESDE S.I.M.O.

En las fotografías de esta página se ve el stand de MICROELECTRÓNICA Y CONTROL S. A. en S.I.M.O. que se celebró en Madrid del 19 al 26 de noviembre. A pesar del ruego que os formulábamos desde SONIMAG, la expectación despertada por el VIC-20 ha sido de las que hacen época, y decíamos lo del ruego para evitar aglomeraciones... Pero no hay manera... ¡En fin! como decimos en Cataluña: cuantos más seamos, más nos reiremos. ¡Un cordial saludo desde S.I.M.O. a todos los lectores de CLUB COMMODORE!

En la fotografía de la izquierda, Jordi Servat, de MICROELECTRÓNICA Y CONTROL, S. A., hace entrega a Javier Meliveo del VIC-20 ganado en el sorteo celebrado en S.I.M.O. ¡Otro afortunado que se incorpora a la amplia comunidad de usuarios! (Fotografías P. Masats)









Depósito Legal Sep. B. 2133-1958 - Altés, s.l., Caballero 87, Barcelona

EDITORIAL

nueva etapa en el Club Commodore

Aquí estamos de nuevo con una significativa mejora: para los suscriptores de CLUB COMMODORE salimos con dieciséis páginas. Sin ánimo de echarnos faroles queremos resaltar un hecho. El "salto" es exactamente el doble (+3 dB). Para dar una cierta idea del esfuerzo que representa para la Redacción de CLUB COMMODORE viene a significar la preparación del DOBLE (+3 dB) de material, o sea que esto no es una simple ampliación sino un gran salto adelante.

Queremos agradecer desde aquí las colaboraciones que ya estamos recibiendo. En este mismo número se inicia una sección dedicada a electrónica a cargo de Rafael Pardo, ya conocido por los lectores de MICRO/BIT. Sólo nos queda repetir el mismo "rollo" de siempre: ¡A VER SI SEGUIMOS EL EJEMPLO!

SE NECESITAN APLICACIONES EDUCATIVAS DE LOS COMMODORE

Aprovechando la ocasión queremos hablar de un tema que consideramos muy importante para los ordenadores personales COMMODORE en general y para el VIC-20 en particular. Se trata de las aplicaciones educativas en sus dos aspectos principales: la enseñanza de la informática propiamente dicha y la ayuda que la microinformática puede prestar a la educación en general. En el primer aspecto, a pesar de nuestros esfuerzos, no hemos detectado muchas iniciativas pero creemos que éstas deben existir. En cuanto a la ayuda que los ordenadores personales pueden prestar a la enseñanza se están haciendo ciertas aplicaciones que son un buen principio pero algunas de ellas, por estar realizadas en el extranjero, son de difícil adaptación en nuestro país. Desde estas páginas queremos hacer un llamamiento a todos aquellos que tengan alguna realización en este campo para que no olviden la plataforma que les ofrece CLUB COMMODORE para dar resonancia a sus trabajos, tanto a nivel de artículos como de reseña de productos existentes.

Sólo nos queda, ya que estamos en diciembre, desear a nuestros lectores, suscriptores, colaboradores y demás -ores y -oras:

¡FELICES FIESTAS
Y PRÓSPERO AÑO NUEVO!
BON NADAL I FELIÇ 1983!
ZORIONAK ETA URTE BERRI ON
¡BOAS FESTAS!

VENTANA CBM

el mapa de memoria CBM

por **JOAN CARLES SAMARANCH**

Con el ánimo de dar cierta continuación a la línea empezada el pasado número con el artículo sobre «overlays» y aprovechando la aparición del mapa de memoria del BASIC 4.0, voy a comentaros, de forma práctica, el funcionamiento interno de las distintas zonas de memoria (ver figura 1).

Página cero y zona de variables del sistema

Por página cero se entiende, estrictamente hablando, la zona comprendida entre la dirección $00 y la $FF (el signo $ significa hexadecimal). De todas formas, el primer Kbyte de memoria, hasta $0400 (1024 en decimal), está reservado para punteros y variables propias del sistema.

Los comentarios sobre estos punteros tengo la intención de irlos documentando con aplicaciones como las de números pasados.

Zona de programas

Esta zona empieza, normalmente, en 1025 ($0401). Por tanto, en las direc-

(pasa a la pág. siguiente)

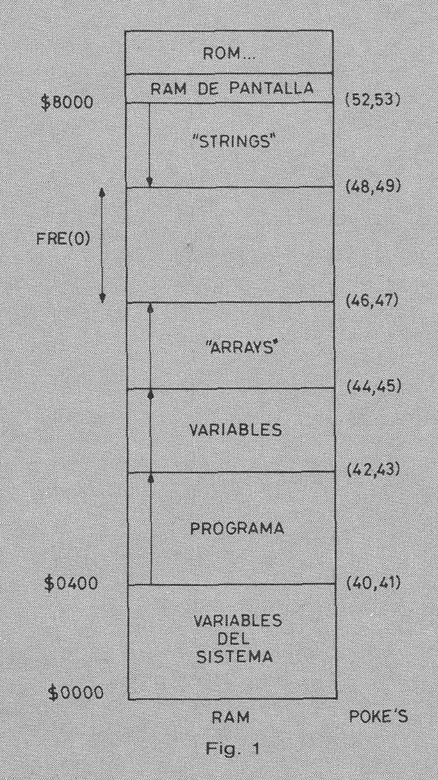


Fig. 1

APLICACIONES

programa para cifrar y descifrar mensajes

LA CODIFICACIÓN DE TEXTOS PARA SU ENVÍO, A TRAVÉS DE UN MEDIO MÁS O MENOS PÚBLICO, DE MANERA QUE SÓLO EL EMISOR Y EL RECEPTOR SEAN CAPACES DE ENTENDERLOS, ES UNA BUENA APLICACIÓN DE LOS ORDENADORES, AUNQUE SUPONEMOS QUE POCOS DE NUESTROS LECTORES SE DEDICAN AL ESPIONAJE, HOY PRESENTAMOS ESTE PROGRAMA



por **P. MASATS**

En la figura podemos ver un programa que permite aplicar el VIC-20 a un tema algo exótico y poco habitual en las revistas serias de informática: el espionaje (del contraespionaje quizá nos ocupemos otro día, del contra-contraespionaje y sucesivos no queremos ni oír hablar). Una de las misiones principales del buen espía es la de comunicar informaciones sin que se entere más gente de la estrictamente necesaria (cosa que hacen espontáneamente los programadores de ordenadores cuando hablan entre ellos). Para ello se utiliza lo que se conoce como una clave, que es una frase corta y fácil de recordar que da una pauta de sustitución de las letras del alfabeto, de manera que los caracteres del mensaje que se ha de transmitir quedan cambiados de la forma que indica la clave. Así si se desconoce ésta, es (teóricamente) imposible descifrar el mensaje.

Vamos a ver un ejemplo. Entramos el programa (extraemos las habituales travesuras de BUG), y pulsamos RUN. A la pregunta «¿TEXTO INICIAL?»: respondemos con el texto que queremos cifrar, que puede ser: «EN UN LUGAR DE LA MANCHA» (original, ¿eh?), pulsamos RETURN y el VIC nos pide: «¿CLAVE SECRETA?» Entramos: «VIC-20 ORDENADOR PERSONAL». Después de un tiempo en la pantalla aparece un ALFABETO DE BASE que es el normal y debajo otro (ALFABETO CIFRADO) que es el resultado de modificarlo en función de la clave. Luego se nos pide si queremos un cifrado o un descifrado. Si pulsamos C obtendremos «TEXTO RESULTANTE»: 2PWJPWNJ VFW2WNV-WAVPCOV que es el resultado del cifrado. A continuación se nos consulta por la opción, cuyas alternativas son:

I: Redefinición de la clave.
D: Descifrado.
C: Cifrado.
F: Fin.
T: Entrar nuevo texto.

Si pulsamos T y volvemos a entrar el texto resultante de la operación anterior **junto con la misma clave** veremos que al descifrar obtenemos el mensaje original. ¡Suerte a los 007 ocasionales!

el mapa de memoria CBM

(viene de la pág. anterior)

ciones 40 y 41 (decimales, por supuesto) figurará esta dirección en el formato peso bajo y peso alto, como podemos comprobar escribiendo en la pantalla:

? PEEK(40)+PEEK(41)*256

La fórmula utilizada convierte en un solo número los dos bytes correspondientes asignándoles el peso adecuado. Lo mismo podemos hacer con las posiciones 42 y 43 que corresponden al final de programa (o principio de la zona de variables), dando el mismo resultado más 2 bytes, debido a los «tres ceros» de fin de programa que algún día explicaremos con más detalle, por tanto: 1027 (en el caso de estar vacía la RAM). Si creamos una línea de programa, por ejemplo:

10 REM**********

el fin de programa estará en 1043, 16 bytes más: 2 para el link de líneas, 2 para el número de línea, 1 para el «token» de la instrucción REM, los 10 correspondientes a los asteriscos y el «cero» de fin de línea. En otro número os comentaré el significado de cada uno de estos términos.

Zona de variables

Llegados a este punto podemos observar, aplicando la anterior fórmula para las posiciones 44 y 45, que el final de la zona de variables tiene como dirección: 1043, longitud cero comparando con el principio de variables. Si escribimos:

X=25

y calculamos otra vez el final de variables nos dará ahora: 1050, 7 bytes más que es lo que ocupa la definición de la cabecera de cada variable. Si añadimos otra variable de otro tipo:

Y%=32766

la longitud se vuelve a incrementar en 7 bytes: 1057. Si variamos el valor de cualquiera de las variables YA definidas no aumenta la longitud de la zona porque no definimos ninguna variable nueva.

Zona de «arrays»

El final de variables coincide con el principio de la zona de «arrays» (matrices), posiciones 44 y 45, y su dirección está almacenada en las posiciones 46 y 47; en este momento igual 1057 (longitud zona de arrays = 0). Si creamos una matriz, por ejemplo:

DIM A%(100)

el fin de la zona de matrices se sitúa en 1266, 209 bytes más: 9 de cabecera y 100 por 2 bytes/elemento por ser entero.

Zona de «strings»

Ésta es la única zona cuyo crecimiento se efectúa al revés de las demás. La dirección de crecimiento viene especificada, en la figura, por las flechas.

Digamos que el principio de la zona de «strings» está en la dirección $8000 (32768, según podemos comprobar aplicando la fórmula para las posiciones 52 y 53), límite con la RAM destinada al almacenamiento de los caracteres que figuran en pantalla.

En estos momentos la longitud de la presente zona es cero (comprobar con las posiciones 48 y 49).

Quizá no hemos dicho, aún, lo que es un «string» pero ha sido adrede para poder explicarlo de forma práctica:

LISTADO DEL PROGRAMA PARA CIFRAR Y DESCIFRAR MENSAJES

```
10 REM+----------------------------+
20 REM+ PROGRAMA PARA CIFRAR Y     +
30 REM+ DESCIFRAR MENSAJES         +
40 REM+ MICROELECTRONICA Y         +
50 REM+ CONTROL; P. MASATS         +
60 REM+ BARCELONA                  +
70 REM+ 14-SEPT.-1982              +
80 REM+----------------------------+
90 PRINT"♥"
100 PRINT"**************************";
110 PRINT"*                        *";
120 PRINT"*  PROGRAMA PARA CI-     *";
130 PRINT"*   FRAR Y DESCIFRAR     *";
140 PRINT"*       MENSAJES         *";
150 PRINT"*                        *";
160 PRINT"**************************";
170 REM
180 AL$="ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123
456789"
190 AL$=AL$+",.'!?;$%#&():*-=<>/@+"
200 AL$=AL$+CHR$(34):REM CHR$(34)="
210 LG=LEN(AL$)
220 REM ENTRADA DEL TEXTO
230 PRINT"TEXTO INICIAL:"
240 INPUTTE$
250 REM INTRODUCCION DE LA CLAVE
260 PRINT"CLAVE SECRETA"
270 INPUTCL$
280 REM SUPRESION DE LAS LETRAS REDUNDAN
TES
290 GOSUB730
300 REM COMPLETAR LA CLAVE CON LAS LETRA
S QUE FALTAN
310 V$=CL$
320 FORN=1TOLG
330 A$=MID$(AL$,N,1)
340 GOSUB820
350 CL$=CL$+A$
360 NEXTN
370 REM IMPRESION DE LOS DOS ALFABETOS
380 PRINT"♥"
390 PRINT"ALFABETO DE BASE:"
400 PRINT
410 PRINTAL$
420 PRINT
430 PRINT"ALFABETO CIFRADO:"
440 PRINT
450 PRINTCL$
460 PRINT
470 REM CIFRADO O DESCIFRADO ?
480 PRINT"CIFRADO O DESCIFRADO
C O D)"
490 INPUTED$
500 IFED$<>"C"ANDED$<>"D"THEN490
510 IFED$="C"THENE$=AL$:S$=CL$:GOTO530
520 E$=CL$:S$=AL$
530 REM BUSQUEDA E IMPRESION DE CADA LET
RA
540 PRINT"♥"
550 PRINT"TEXTO RESULTANTE:":PRINT
560 V$=E$
570 FORL=1TOLEN(TE$)
580 A$=MID$(TE$,L,1)
590 GOSUB820
600 IFA=0THENPRINTA$;:GOTO620
610 PRINTMID$(S$,A,1);
620 NEXTL
630 PRINT
640 REM SEQUIR
650 PRINT
660 PRINT"FUNCION (I,D,C,F O T)"
670 INPUTED$
680 IFED$="D"ORED$="C"THEN500
690 IFED$="I"THENPRINT"♥":GOTO250
700 IFED$="T"THENPRINT"♥":GOTO220
710 IFED$<>"F"THENPRINT"♥":GOTO660
720 END
730 REM SUPRESION DE LETRAS REDUNDANTES
EN LA CLAVE
740 C$=""
750 FORN=1TOLEN(CL$)
760 A$=MID$(CL$,N,1)
770 V$=C$:GOSUB830
780 C$=C$+A$
790 NEXTN
800 CL$=C$
810 RETURN
820 REM BUSQUEDA DE CARACTER DENTRO DE U
NA CADENA
830 A=0
840 FORK=1TOLEN(V$)
850 IFA$=MID$(V$,K,1)THENA$="":A=K:K=LEN
(V$)
860 NEXTK
870 RETURN

READY.
```

C$=«COMMODORE»

esto es: una cadena (secuencia) de caracteres. El número máximo de caracteres que podemos darle a una cadena es de 255 bytes.

Volviendo al tema de las «zonas», sin darnos cuenta acabamos de crear una cadena de 9 caracteres, que junto a los 2 bytes de lincaje (enlace) necesarios hacen un total de 11 bytes: 32768 — 11 = 32757, que es la dirección actual del fin de la zona de «strings».

Las instrucciones CLR y NEW

Una vez vistas, por encima, las distintas zonas de memoria en que se divide la RAM con el trabajo del BASIC de Commodore, supongo tenéis una idea un «poco» más clara de su funcionamiento.

Y hablando de «clara» vamos a limpiar las distintas zonas de variables, en general, con la instrucción CLR (clear). Lo que hace este «statement» del BASIC es, sencillamente poner a cero las longitudes de estas zonas haciendo las posiciones de fin de arrays y fin de variables iguales a la dirección del principio de variables. Como si hiciéramos la siguiente secuencia:

```
POKE44,PEEK(42):POKE46,PEEK(42)
POKE45,PEEK(43):POKE47,PEEK(43)
```

Algo parecido ocurre con la zona de «strings», ya que también queda a cero.

La instrucción NEW la única novedad que incorpora es que, además, se «come» el programa, haciendo la dirección de fin de programa igual a la de inicio. ¡PERO NO LO MACHACAN!

Os recomiendo que comprobéis todos estos pasos.

EL futuro en Ventana CBM

Una vez abierta la polémica, en el presente número, de forma muy superficial, pero a la vez fácil de entender, tengo la intención de profundizar más en esta línea con los temas que provisionalmente titulo:

- Zona de strings 2.ª parte y el «garbage collection»
- Viaje a las profundidades de la zona de programas
- Estructura de las cabeceras de variables

Como siempre, acepto vuestras sugerencias y me gustaría saber si preferís que a todo esto se adelante el «cursillo» acerca del funcionamiento de las unidades de discos «Commodore», o no.

```
10 REM EMISOR COMUN
20 PRINT"♥"TAB(3)"EMISOR COMUN"
30 PRINT:PRINT"1-NPN"
40 PRINT:PRINT"2-PNP"
50 PRINT:PRINT"PULSE 1 O 2"
60 GETZ$:IFZ$="1"THENN=1:GOTO90
70 IFZ$="2"THENN=2:GOTO90
80 GOTO60
90 PRINT"♥":INPUT"UCC(V)";UC
100 PRINT:INPUT"IC(MA)";IC
110 PRINT:INPUT"UCE(V)";U1
120 PRINT:INPUT"UE(V)";U2
130 PRINT:INPUT"HFE";HF
140 PRINT:INPUT"S";S
150 PRINT:INPUT"HIE(KOHMS)";HI
160 PRINT:INPUT"RG(KOHMS)";RG
170 PRINT:INPUT"RL(KOHMS)";RL
180 PRINT:INPUT"FTMIN(HZ)";FT
190 PRINT"♥"
200 IB=IC/HF:IE=IB+IC:RE=U2/IE
210 RC=(UC-(U2+U1))/IC
220 RB=((RE*S)-RE)/(1-(S/(HF+1)))
230 IFN=1THENUH=IB*RB+.6+IB*RE*(HF+1):GO
TO250
240 UH=IB*RB-.6+IB*RE*(HF+1)
250 R1=UC*(RB/UH):R2=(RB*R1)/(R1-RB)
260 PRINT"CALCULO DE LAS RESISTENCIAS"
270 PRINT:PRINT"RC=";RC;"KOHMS"
280 PRINT:PRINT"RE=";RE;"KOHMS"
290 PRINT:PRINT"R1=";R1;"KOHMS"
300 PRINT:PRINT"R2=";R2;"KOHMS"
310 GOSUB830
320 PRINT"♥**NORMALIZAR**"
330 PRINT"RC=";RC:PRINT"RE=";RE:PRINT"R1
=";R1:PRINT"R2=";R2
340 PRINT:PRINT"NORMALICE ESTOS VALORES"
350 PRINT:INPUT"RC";RZ:INPUT"RE";RN:INPU
T"R1";R3:INPUT"R2";R4:PRINT"♥"
360 RV=(R3*R4)/(R3+R4):US=(UC*R4)/(R3+R4
)
370 IFN=1THENIV=(US-.6)/(RV+(RN*(HF+1)))
:GOTO390
380 IV=(US+.6)/(RV+(RN*(HF+1)))
390 IZ=HF*IV:IN=IV+IZ
400 U4=IN*RN:UB=IZ*RZ
410 UD=UC-(UB+U4)
420 PRINT"VARIACION DEL PUNTO Q"
430 PRINT"IC(DADA)=";IC
440 PRINT"IC(CALCULADA)=";IZ
450 PRINT"UCE(DADA)=";U1
460 PRINT"UCE(CALCULADA)=";UD
470 PRINT"UE(DADA)=";U2
480 PRINT"UE(CALCULADA)=";U4
490 PRINT:PRINT"1-VARIACION DE LOS DATOS
"
500 PRINT:PRINT"2-SEGUIR EL PROYECTO"
510 PRINT:PRINT"PULSE 1 O 2"
520 GETB$:IFB$="1"THEN20
530 IFB$="2"THEN550
540 GOTO520
550 PRINT"♥":GM=(IZ/26)*1000
560 R5=HF/GM:R6=HI-R5
570 UT=RV/(RV+RG)
580 Y=(RZ*RL)/(RZ+RL)
590 RT=(RV*RG)/(RV+RG)
600 D=(RT/(RT+HI))*R5*GM*Y
610 AV=20*(LOG(D)/LOG(10))
620 PRINT"CALCULO DE LA GANANCIA"
630 PRINT:PRINT"AV=";AV;"DB"
640 GOSUB830
650 PRINT"♥CALCULO DE LOS CONDENSADORES"
660 L=R6+R5+RN*(HF+1):RH=RG+((RV*L)/(RV+
L))
670 W=(R5+R6+RV)/(HF+1):RM=(RN*W)/(RN+W)
680 RO=RZ+RL
690 CI=1E+3/(2*π*RH*(FT/5))
700 CE=1E+3/(2*π*RM*FT)
710 CO=1E+3/(2*π*RO*(FT/5))
720 PRINT:PRINT"CIN=";CI;"UF"
730 PRINT:PRINT"CE=";CE;"UF"
740 PRINT:PRINT"COUT=";CO;"UF"
750 GOSUB830
760 PRINT"♥1-REPETIR RESULTADOS"
770 PRINT:PRINT"2-NUEVO PROYECTO"
780 PRINT:PRINT"3-FIN DE SESION DE TRABA
JO"
790 PRINT:PRINT"PULSE 1,2 O 3"
800 GETE$:IFE$=""THEN800
810 Z=VAL(E$):ONZGOTO190,20,820
820 END
830 PRINT:PRINT"PARA SEGUIR PULSE CUALQU
IER TECLA"
840 GETA$:IFA$=""THEN840
850 RETURN

READY.
```

(Fig. 1)



Aquí empieza una colaboración que esperamos sea interesante para los lectores de CLUB COMMODORE. R. Pardo, que ya ha publicado algunos trabajos en «MICRO/BIT», va a llevar sobre sus hombros la responsabilidad de una sección de nuestra Revista dedicada a programas para diseño de electrónica. ¡Le deseamos mucha suerte!

Este programa calcula etapas preamplificadoras de baja frecuencia en emisor común, con transistores PNP o NPN de silicio. Si se quisiera trabajar con transistores de germanio se debería cambiar en las líneas 230, 240, 370 y 380 el valor .6 por el .1.

El programa funciona de la siguiente manera:

Las líneas 20-80 piden el tipo de transistor con el que se quiere trabajar (NPN-PNP) y se debe pulsar solo la tecla que el ordenador indica (no es necesario pulsar RETURN).

Las líneas 90-180 permiten entrar los valores de los parámetros a los que se debe ceñir el diseño. (Se deben introducir los valores en las unidades que le indique el programa. No debe hacerse de otra manera porque podría dar resultados erróneos). Después de introducir cada valor se debe pulsar RETURN.

En las líneas 190-250, el programa

VARIABLE	NPN	PNP
Ucc	20 V.	20 V.
Ic	2 mA.	2 mA.
Uce	4 V.	4 V.
Ue	1 V.	1 V.
hFE	100	100
S	10	10
hie	6 Kohms.	6 Kohms.
Rg	1 Kohms.	1 Kohms.
RL	15 Kohms.	15 Kohms.
Ft	100 Hz.	100 Hz.
Rc	7.5 Kohms.	7.5 Kohms.
Re	.495 Kohms.	.49 Kohms.

VARIABLE	NPN	PNP
R1	58.214 Kohms.	198.23 Kohms.
R2	5.404 Kohms.	5.07 Kohms.
Rc(Norm.)	7.5 K.	7.5 K.
Re(Norm.)	.51 K.	.51 K.
R1(Norm.)	56 K.	200 K.
R2(Norm.)	5.1 K.	5.1 K.
Ic(Cal.)	1.903 mA.	1.942 mA.
Uce('')	4.744 V.	4.138 V.
Ue('')	.9809 V	1.000 V.
Av(dB)	35.61 dB.	35.69 dB.
Cin	1.494 muF.	1.426 muF.
Ce	18.179 muF.	17.769 muF.
Cout	.353 muF.	.353 muF.

cálculo de etapas en emisor común

por **R. PARDO**

calcula los valores de las resistencias de polarización y en las líneas 260-300 los presenta. Si alguno de los resultados da negativo es debido a que antes se han introducido unos valores que no «cuadran» con los demás. Revise sus datos, corríjalos y vuelva a ejecutar el programa.

Las líneas 320-350 piden que se normalicen los valores de resistencia. Las líneas 360-410 calculan la variación del punto Q; las líneas 420-450 le presentan las variaciones de Q que han ocurrido y pregunta si debe seguir el diseño o vuelve a presentar la entrada de datos para corregir alguno de ellos. Si se elige lo último, no deben introducirse de nuevo todos los valores. Púlsese RETURN hasta que aparezca el que se quiere variar, introdúzcase el valor y repítase el RETURN hasta que se termine de ejecutar la entrada de datos. Entonces el programa sigue normalmente.

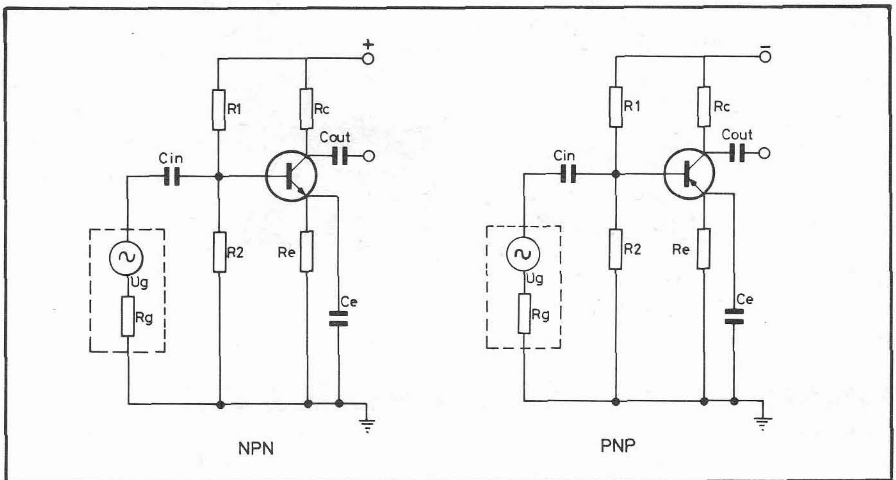
Sigue el programa y las líneas 550-610 calculan la ganancia total de la etapa, presentándola en las líneas 620-630. El cálculo de los condensadores de acoplo se realiza en las líneas 650-710, presentándose sus valores en microfaradios (líneas 720-740).

Las líneas 760-820 preguntan si se desea iniciar un nuevo proyecto, repetir resultados (en este caso, en la sección NORMALIZAR no debe volverse a introducir de nuevo los valores normalizados; bastará con pulsar RETURN cuatro veces y el programa presentará los valores anteriores de la variación del Q) o bien finalizar la ejecución del programa en cuyo caso el ordenador responderá READY.

Las líneas 830-850 contienen un pequeño subprograma para facilitar el paso de una pantalla a otra.

JUEGOS DE VALORES PARA LA COMPROBACIÓN DEL PROGRAMA

En la tabla adjunta se dan dos juegos de valores para la comprobación del programa. ■



RADIOAFICIÓN

inicios del VIC-20 en RTTY

Sabemos de algunos radioaficionados que ya están trabajando la modalidad de RTTY con el VIC-20, pero también sabemos de «otros» que solamente estamos al inicio del tema.

Por eso cuando cayó en nuestras manos una revista alemana, con un programa titulado TELEX, un circuito y las palabras Baudot y TTY, cundió el pánico. ¿Será?, ¿no será? Pusimos manos a la obra: entrar el programa y buscar al amigo de turno que nos tradujera, dentro de lo posible, aquella jerga.

El resultado de la traducción indicaba que, para los que deseasen utilizar como impresora barata del VIC-20 un teletipo, aquel programa traducía los caracteres y, con el pequeño circuito de la figura 1, lo acoplábamos. Si traducía los caracteres para el teletipo, lo podíamos utilizar ¡en RTTY! Había que comprobarlo.

El funcionamiento del programa indicaba que cualquier mensaje que imprimiésemos por pantalla o impresora saldría también hacia el teletipo.

La primera prueba que se hizo fue imprimiendo varias «R», que conocíamos de verlas en diagramas de revistas, y observando la señal en el osciloscopio. Tenía un bit de arranque, los cinco de mensaje correspondientes, uno y medio de stop y cada uno duraba 22 ms: ¡era BAUDOT!

Después de este primer éxito de-

(pasa a la pág. siguiente)

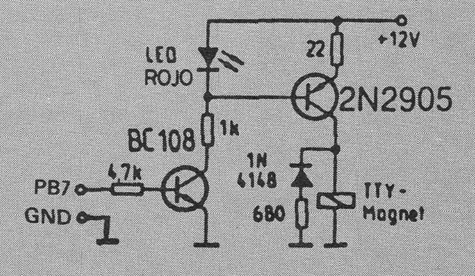


Fig. 1 - Esquema de conexión del teletipo con el port del usuario.

RINCÓN DEL VICCOLAGE



cómo au su hogar con el

Alguno de nuestros amigos se estarán preguntando si en verdad el equipo de Redacción está descuidando un poco su ortografía, otros dirán dirán que nuestra mascota, el BUG, ya ha estado haciendo de las suyas; bien, pues ni lo uno ni lo otro. El nombre de «RINCÓN DEL VICCOLAGE», con el que, a partir de este momento y hasta que no se nos ocurra otro mejor, denominaremos a esta sección de la Revista, intentará ser una fuente de ideas luminosas para esa rara especie — algunos dicen que en vías de extinción — llamada HOMO VICOLATRIX, especie ésta que se diferencia del HOMO SAPIENS en que se empeña en hacer él mismo lo que los demás saben hacer mejor y más rápido (y a los que ha de recurrir al final en muchos casos).

En esta sección intentaremos dar ideas simples, fáciles de poner en práctica y que no signifiquen un gran desembolso para nuestros lectores. En suma, ideas geniales.

La idea general de los artículos que en esta sección irán apareciendo, es

inicios del VIC-20 en RTTY

(viene de la pág. anterior)

```
1 POKE56,29
5 PRINT"□           TELEX"
10 FORX=0TO255:READA:POKE7424+X,A:NEXT
20 PRINT"■SYS7648--IMPRESION ON"
30 PRINT"SYS7659--IMPRESION OFF":END
100 DATA165,249,162,128,142,18,145,201,1
3,208,16,169,8,32,33,29
116 DATA169,2,32,33,29,169,0,76,33,29,23
4,41,63,170,189,116
132 DATA 29,133,249,41,32,197,248,240,14
,133,248,168,240,4,169,27
148 DATA208,2,169,31,32,63,29,165,249,32
,63,29,198,250,96,162
164 DATA127,142,16,145,32,102,29,160,4,7
4,144,4,162,255,208,2
180 DATA162,127,142,16,145,32,102,29,136
,16,238,162,255,142,16,145
196 DATA162,30,32,104,29,96,162,20,134,2
47,162,223,202,208,253,198
212 DATA247,16,247,96,0,3,25,14,9,1,13,2
6,20,6,11,15
228 DATA18,28,12,24,22,23,10,5,16,7,30,1
9,29,21,17,47
244 DATA0,50,45,0,36,60,37,45,5,61,49,37
,47,50,45,49
260 DATA44,35,60,61,54,55,51,33,42,48,53
,39,38,56,46,44
276 DATA36,62,36,57,72,133,249,138,72,15
2,72,165,249,201,13,240
292 DATA14,166,250,208,16,72,169,13,234,
133,249,32,0,29,104,162
308 DATA65,134,250,133,249,32,0,29,104,1
68,104,170,104,76,122,242
324 DATA169,180,141,38,3,169,29,141,39,3
,96,169,122,141,38,3
340 DATA169,242,141,39,3,96,234,234,234,
234,234,234,234,234,234,234

READY.
```

(Fig. 2)

cidimos la comprobación de todos los caracteres pero con otro método: en AFSK via radio.

Se montó el modulador que pareció más idóneo: con un solo circuito integrado, el XR-2206, directamente acoplable al port del usuario.

Otro colega con mucha paciencia aguantó el recital de pitos comprobando todos los caracteres: ¡FUNCIONABA!

A posteriori hemos procedido al análisis del programa (fig. 2), de sólo 256 bytes en código máquina. La base de funcionamiento consiste en interferir la rutina de impresión, traduciendo cada carácter a Baudot y dando la señal a través del port del usuario antes de imprimir.

Una vez cargado el programa y ejecutado el RUN, lo único que ha pasado es que hemos cargado y protegido el programa en código máquina contenido en las líneas DATA. Para activarlo tecleamos SYS7648 y para desactivarlo con SYS7659 o bien pulsando las teclas RUN y RESTORE simultáneamente.

La primera prueba que podemos efectuar es activar el invento y hacer un LIST. La velocidad del listado se produce a 45.45 baudios por la salida PB7 del port del usuario con los caracteres correspondientes en Baudot.

La salida está a 5 voltios («1» lógico) en reposo (espacio) y a cero cuando se produce una marca.

Otra curiosidad es el salto automático de línea cuando llega a la columna 65. Esto es muy interesante para los que nos reciben con un teletipo.

Como editor lo más sencillo es, después de haber cargado y ejecutado el programa TELEX, utilizar:

```
10 SYS 7648
20 GET A$:IF A$=""THEN20
30 PRINT A$;:GOTO20
```

En otra ocasión os comentaremos el editor utilizado actualmente y el esquema del modulador de AFSK, así como la parte receptora. ■

tomatizar

VIC-20

por **M. SANS**

la de poder disponer de una serie de desarrollos simples y modulares que nos permitan, en un momento dado, recopilarlos parcial o totalmente en una aplicación determinada, por ejemplo, CÓMO AUTOMATIZAR SU HOGAR CON EL VIC-20, aplicación ésta que será la base de nuestros próximos artículos.

Hacemos una invitación desde estas líneas, para que todos aquellos que deseen ver su nombre y obra impresos en letras de molde, nos remitan sus trabajos lo más completos, detallados e inteligibles posible. Los trabajos tendrán que ser comprobados, por lo cual sería de agradecer que, junto con el artículo, nos remitieran un prototipo del desarrollo, que será devuelto.

La manera más elegante de presentarnos un artículo sería:

1. El artículo en sí, con una justificación lo más clara posible del porqué del desarrollo; una descripción a nivel de señales del funcionamiento; una lista de materiales, etcétera.
2. En folios aparte, los esquemas del diseño en cuestión. ¿Entendido? Bien, pues ánimo y adelante, esperamos vuestros trabajos.

Sí, sí, todo esto está muy bien — dirá alguno de nuestros sufridos lectores —, pero ¿qué hacemos los que sin dedicarnos al VICCOLAGE tenemos algún problema de HARDWARE? ¿No podríamos solucionarlo nosotros mismos? ¿Será esto VICCOLAGE?

Bien, tranquilos, los amigos que tengan algún problemita pueden escribirnos con el máximo de detalles e intentaremos solucionarlo, publicando la respuesta en otra sección de la Revista, a la que llamaremos ¡SOCORRO!

UNA APLICACIÓN PARA AHORRAR ENERGÍA

En estos tiempos en que la energía es tema cotidiano, veamos qué es lo que podemos hacer para aprovecharla mejor.

Con la entrada en el invierno, es cada vez más común la utilización de calefacción eléctrica, bien en forma de placas, bien en forma de radiadores o calefactores (¿dónde están los braseros o las chimeneas con sus troncos ardiendo?). Bien, pues, la aplicación que proponemos nos permitirá ahorrar en calefacción del orden del 10 al 20 %.

La primera pregunta que se nos ocurre es ¿cómo podemos utilizar nuestro VIC-20 para controlar la calefacción?, ¿qué interfaz supercomplicada tendremos que diseñar para controlar todo esto? Bueno, la primera idea que se nos ocurre consiste en disminuir la calefacción durante las horas de sueño. ¡Bravo!, ya tenemos un punto de partida. Si, además, disminuimos también la temperatura, dependiendo de la ausencia de ocupantes en la habitación, tendremos un ahorro considerable.

Bien, tomemos como ejemplo un radiador de convección (placa) con un termostato incorporado. El sistema de control que hemos de adoptar ha de ser tal que no necesitemos de una interfaz de potencia, ni tengamos que efectuar grandes modificaciones en nuestra instalación eléctrica. Además, en caso de fallo del ordenador, cosa no del todo improbable (¡toquemos madera!) hemos de evitar, en lo posible, que los ocupantes de la habitación adquieran la apariencia de ¡pollos a l'ast! La solución que se nos ha ocurrido es poner cerca del termostato una resistencia de tal manera que podamos hacer que radíe calor, con lo cual el termostato creerá que la temperatura de la habitación es más alta, y la disminuirá. Esta resistencia se ha de colocar lo más cerca posible del bimetal del termostato.

(pasa a la pág. siguiente)

cómo automatizar su hogar con el VIC-20

(viene de la pág. anterior)

En la figura 1 tenemos la estructura general del conjunto resistencia-controlador-radiador.

En la figura 2 se da el diagrama del circuito de control en el que, mediante un circuito CMOS 4049, controlamos un transistor trabajando en conmutación. Con este transistor (un 2N3643 ó equivalente) controlamos un relé reed de 5 voltios.

Si desde nuestro VIC-20 programamos un nivel bajo en la salida PB0 del port del usuario, el circuito IC1 nos aplicará un nivel alto a la base del transistor TR1, activando éste al relé.

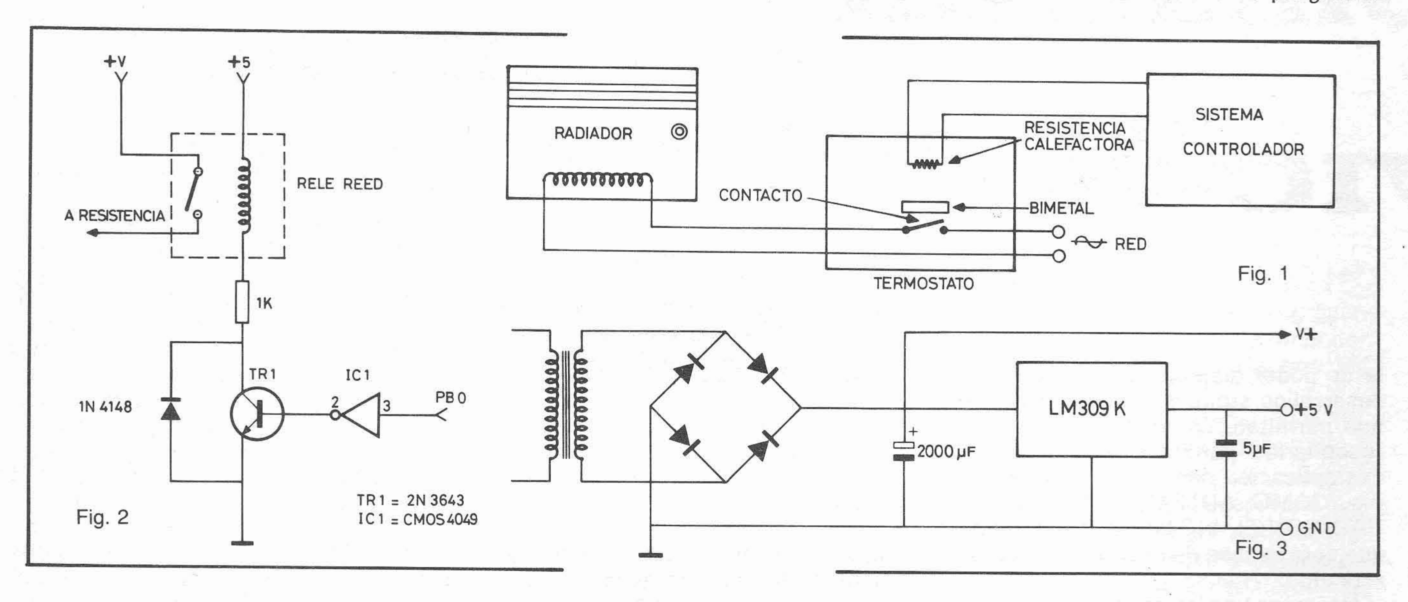


Fig. 2

Fig. 1

Fig. 3

Vemos que, en el caso de un fallo en el controlador o en el circuito de interfaz, en el peor de los casos (que el relé se quedase activado) la temperatura de la habitación tendería a bajar.

En la figura 3 se presenta el diagrama de la fuente de alimentación, ya que tenemos un consumo limitado en el VIC. En el próximo número veremos cómo controlar el número de personas en una habitación y el programa de control del sistema en BASIC. ■

MEA CULPA

una aclaración al listado del artículo "DOS PROGRAMAS INTERESANTES"

(Publicado en el n.° 2, pág. 7)

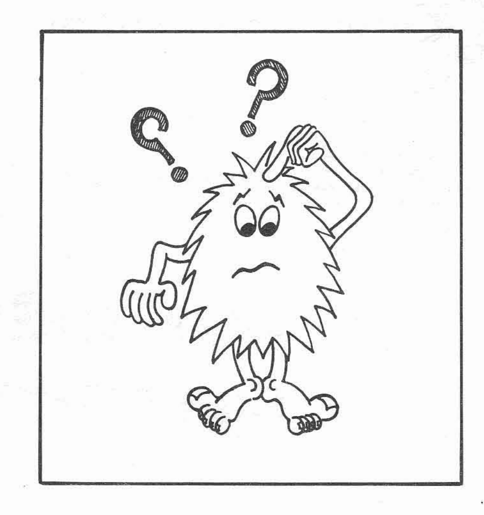
```
25 print chr$(14) "■     PARA-LADRILLOS",
"                    "
26 print "■■Debes IMPEDIR que se  te amo
ntonen los...",,,"          LADRILLOS"
27 print "■Para conseguirlo...:",,,"
  PARALOS!!!"
28 print "■*Control de la barra:",,," ´,
´......Izquierda."
29 print "■ ´.´........Derecha."
30 print "■ **Para comenzar:",,," PULSA
UNA TECLA."

ready.
```

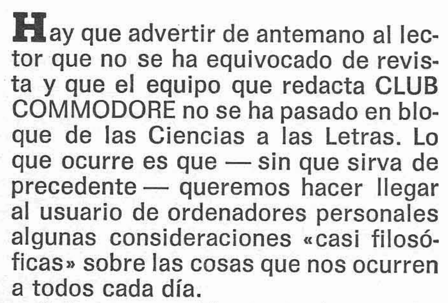
Sería por la emoción de publicar nuestra primera colaboración, o porque estábamos en la Luna (no solemos ir a menudo), pero la cuestión es que se nos coló una mala pasada de BUG en el listado de la izquierda de J. MELGAR. En las líneas 25, 26, 27, 28, 29 y 30 hay una serie de mensajes en minúsculas que reconocemos son difíciles de interpretar tal como los hemos listado. Para reparar el error y después de pedir los perdones que haga falta (y habiendo castigado a BUG de cara a la pared durante algunos milisegundos) damos en la figura adjunta el listado correspondiente en minúsculas.

FILOSOFÍA APLICADA (I)

la ley de Murphy y sus corolarios



por **PERE MASATS**



Hay que advertir de antemano al lector que no se ha equivocado de revista y que el equipo que redacta CLUB COMMODORE no se ha pasado en bloque de las Ciencias a las Letras. Lo que ocurre es que — sin que sirva de precedente — queremos hacer llegar al usuario de ordenadores personales algunas consideraciones «casi filosóficas» sobre las cosas que nos ocurren a todos cada día.

Solemos considerar que hoy en día la Ciencia tiene una respuesta para cada pregunta que podamos formular. Aunque esto no es comúnmente aceptado por los científicos profesionales, al hombre de la calle se lo han hecho creer de una manera o de otra y en unos tiempos en los que se tiende a valorar las cosas que nos rodean desde un punto de vista estrictamente práctico, no dejan de sorprender ciertos «comportamientos» de objetos supuestamente inanimados y la malévola inoportunidad de ciertas «casualidades».

Si usted ha recibido alguna vez una llamada telefónica en el momento de instalarse en el baño; si después de esperar un buen rato el autobús éste aparece a los pocos segundos de encender un cigarrillo; si usted ha lavado el automóvil horas antes de que lloviera o le ha sorprendido la lluvia el único día que no llevaba paraguas, entenderá perfectamente a qué nos referimos. (Conscientemente evitamos hacer una lista — siquiera superficial — de los centenares de trastadas que sufrimos habitualmente los usuarios de ordenadores. Sería demasiado fácil.)

La Ley de Murphy fue el primer paso que dio un ingeniero aeronáutico norteamericano de este nombre en 1949 para «sistematizar» (?) estos imponderables que, por serlo, escapan a la Ciencia y — es evidente — no pueden ser tratados seriamente ni por la Lógica ni la Mística. La Ley de Murphy dice:

SI ALGO PUEDE IR MAL, IRÁ MAL

Lo cual no parece gran cosa a primera vista... Hasta que uno — naturalmente escéptico — empieza a ensayar su aplicación. Entonces, los resultados son pasmosos.

Desde 1949 ha corrido mucha tinta y en lo que respecta a los aspectos prácticos de la Ciencia las cosas han avanzado a una velocidad no menor a la causada por una enérgica patada en el trasero. Con ello, la Técnica también se ha complicado y con ella han crecido las oportunidades de la aplicación de esta Ley. También con el paso del tiempo se han multiplicado los filósofos espontáneos y con ellos los corolarios y adaptaciones a diferentes aspectos del trabajo de cada día. En este artículo vamos a dar una lista de los que han llegado a nuestras manos — gracias a la colaboración de M. Amado de «Microelectrónica y Control» —, y como no sufrimos el síndrome de la falsa modestia empezaremos por nuestra versión de la **Ley de Murphy:**

SI ALGO PUEDE IR MAL, NO SÓLO IRÁ MAL SINO QUE ADEMÁS LO HARÁ EN EL MOMENTO MÁS INOPORTUNO

Así, el llamado COMITÉ DE LA SOCIEDAD INTERNACIONAL DE INGENIEROS FILÓSOFOS dice haber descubierto lo siguiente:

1. En cualquier fórmula las constantes (especialmente las obtenidas en manuales de ingeniería) deben ser tratadas como variables.
2. El prometido plazo de entrega debe multiplicarse por un factor de 2.0.
3. El rendimiento estimado por el fabricante de un producto debe multiplicarse por 0.5.
4. El rendimiento estimado por el vendedor debe multiplicarse por un factor de 0.25.
5. Cuando más de una persona es responsable de un error de cálculo, ninguna tendrá la culpa.
6. Cuando se establezca un índice de seguridad, un idiota ingenioso calculará rápidamente un método para excederlo.
7. Las cláusulas de garantía quedan anuladas con el pago de la factura.
8. La pieza que la fábrica se ha olvidado enviar es la única absolutamente necesaria para que el aparato funcione.
9. No sólo la fábrica se ha olvidado enviar la pieza sino que además raramente la fabrica.
10. Las partes que precisen ajuste o servicio periódico serán las menos accesibles.

Como piezas sueltas de este muestrario veamos las siguientes:

- Es imposible hacer nada a prueba de tontos porque los tontos son muy ingeniosos.
- La otra cola siempre avanza más aprisa.
- Cuando nuestro avión llega con retraso, el avión con el que debemos enlazar despegará puntualmente.
- Las posibilidades de que una tostada caiga al suelo por el lado de la mantequilla son directamente proporcionales al precio de la alfombra que hay debajo.
- Cualquier aparato funciona mejor cuando se enchufa.
- No hay que forzar nunca un aparato; basta con utilizar un martillo más grande.
- En tareas de investigación, si los hechos contradicen la teoría, los hechos deben ser descartados de inmediato.
- El hombre que sonríe cuando las cosas van mal, es que ya ha encontrado a quien echarle la culpa.
- Bajo las condiciones más rigurosamente controladas de presión, temperatura, volumen, humedad y demás variables, el organismo reaccionará como le venga en gana.
- En póker, una pistola Parabellum gana a cuatro ases.

(continuará)

MAPA DE MEMORIA DEL VIC-20 (IV)

un editor de caracteres para el VIC

por **PERE MASATS**

Este es — de hecho — la continuación lógica del artículo anterior, en el que se realizaba un análisis detallado de las funciones del «chip» más importante del VIC-20: el controlador de video. A causa de la naturaleza del tema no era posible dar una aplicación inmediata al contenido del artículo y éste tuvo que «salir» con más texto que listado. En CLUB COMMODORE es nuestra intención dar — siempre que sea posible — la información teórica necesaria pero acompañada de una aplicación práctica inmediata y probada. Éste es el caso del programa que presentamos en este número y que es uno (el mejor a nuestro juicio) de los conocidos como «Editores de caracteres». ¿Qué es un «Editor»? Como este concepto se utiliza a menudo en informática, vamos a detenernos un poco en él.

Un «Editor» es un programa que realiza unas funciones específicas. Las principales pueden ser: entrar datos, ordenarlos de alguna manera, modificarlos y, por último, guardarlos para posterior utilización por el mismo u otro programa. Si aplicamos esto al caso de un editor de caracteres para el VIC, ello quiere decir que podremos crear nuevos caracteres, modificar los existentes y, de alguna manera, guardarlos en disco o cassette para poder utilizarlos en los programas que los necesiten.

Si entramos el programa cuyo listado se da en la figura 1 y lo ponemos en marcha, después del título nos dará a entender que debemos esperar a que termine sus esotéricas actividades. Luego aparecerá la pantalla dividida en tres áreas: abajo, a la izquierda, se exhiben los caracteres que se pueden modificar; arriba, a la izquierda, los cuatro caracteres que estamos modificando (se suele decir también «editando»); en este momento y, a la dere-

```
10 PRINT"■ EDITOR DE CARACTERES"
20 PRINT"■    PARA EL VIC-20"
30 FORN=1TO22:PRINT"¯";:NEXT
50 PRINT"■PULSE ■RETURN■ PARA      ■
EMPEZAR"
60 GETA$:IF A$<>CHR$(13)THEN60
70 PRINT"■UN MOMENTO POR FAVOR":GOSUB38
0
100 POKE36869,255:POKE56,26:POKE55,0:POK
E650,255:POKE52,26:V=7168
120 GOSUB360:X=0:Y=0
130 GOSUB390
140 GOSUB160
150 GOTO130
160 GETA$:A=62-PEEK(37137):POKE37154,127
:B=PEEK(37152):POKE37154,255
190 IFA$="Q"ORA$="W"ORA$="E"ORAAND4THENY
=Y-1
200 IFA$="Z"ORA$="X"ORA$="C"ORAAND8THENY
=Y+1
210 IFA$="Q"ORA$="A"ORA$="Z"ORAAND16THEN
X=X-1
220 IFA$="E"ORA$="D"ORA$="C"ORB=119THENX
=X+1
230 IFAAND32ORRQTHENA$=CHR$(133)
240 IFX=-1THENX=15
250 IFX=16THENX=0
260 IFA$=CHR$(133)THENPOKEP-30720,170:GO
SUB510:POKEQ,PEEK(Q)OR2↑(7-RX)
270 IFA$=CHR$(134)THENPOKEP-30720,174:GO
SUB510:POKEQ,PEEK(Q)AND(255-2↑(7-RX))
280 IFY=-1THENY=15
290 IFY=16THENY=0
300 IFA$=CHR$(135)THENGOSUB410
310 IFA$=CHR$(136)THENGOSUB550
320 IFA$=CHR$(137)THENA=PEEK(36869):POKE
36869,255:IFA=255THENPOKE36869,240
330 IFA$=CHR$(138)THEN580
340 IFA$=CHR$(139)THEN800
350 RETURN
360 PRINT"■":GOSUB490:FORY=19TO23:FORX=0
TO15:POKE7680+Y*22+X,X+(Y-19)*16
370 POKE38400+22*Y+X,2:NEXTX,Y:A$="@":GO
SUB420:RETURN
380 FORN=0TO64*8:POKE7168+N,PEEK(32768+N
):NEXT:RETURN
390 P=38422+X+Y*22:POKEP,7:IFP<>STHENPOK
ES,4
400 S=P:RETURN
410 PRINT"■?":GETA$:IFA$=""THEN410
420 PRINT"■ ■"A$" ■  ":CH=PEE
K(7764):IFCH>47THEN410
430 GOSUB490:H=7764:POKEH+1,CH+1:POKEH+2
2,CH+16:POKEH+23,CH+17
440 FORY2=0TO1:FORX2=0TO1:CI=CH+X2+Y2*16
450 FORY1=0TO7:FORX1=0TO7
460 IFPEEK(V+CI*8+Y1)AND(2↑(7-X1))THENPO
KE7702+(Y1+Y2*8)*22+X1+X2*8,170
470 NEXTX1,Y1,X2,Y2
480 RETURN
490 PRINT"■":FORN=1TO16:PRINT"■........
........":NEXT
500 D=174:RETURN
510 CI=CH:RX=X:RY=Y:IFX>7THENCI=CH+1:RX=
RX-8
520 IFY>7THENCI=CI+16:RY=RY-8
530 Q=V+CI*8+RY
540 RETURN
550 FORN=V+CH*8TO7175+(CH+1)*8:POKEN,0:N
EXT:GOSUB490
560 FORN=V+(CH+16)*8TO7175+(CH+17)*8:POK
EN,0:NEXT:RETURN
580 POKE36869,240:NO=9000+CH*10
600 PRINT"■":FORCH=CHTOCH+16STEP16
610 F=V+CH*8
620 PRINTNO"DATA"F",";:FORN=FTOF+15:PRIN
TMID$(STR$(PEEK(N)),2)",";:NEXT:PRINT"■
"
640 NO=NO+5:NEXTCH
650 PRINT"RUN100":PRINT"■";
700 PRINT"■";:POKE198,9:FORN=631TO640:PO
KEN,13:NEXT:END
800 N=0
810 PRINT"■":FORM=NTON+60STEP10:PRINTM
:NEXT:PRINT"800N="M:PRINT"RUN800"
820 IFN<660THEN700
830 PRINT"■100READN:IFNTHENFORN=NTON+1
5:READA:POKEN,A:NEXT:GOTO100"
840 PRINT700:FORN=800TO840STEP10:PRINTN:
NEXT:PRINT"POKE1,0:SYS1":GOTO700
9999 DATA 0

READY.
```

Fig. 1

ÉSTE ES UN PROGRAMA QUE NOS PERMITE DEFINIR NUESTROS PROPIOS JUEGOS DE CARACTERES DE UNA MANERA FÁCIL Y PRÁCTICA Y ASÍ APROVECHAR LAS POSIBILIDADES GRÁFICAS DEL EQUIPO

cha, el aspecto en tamaño natural de los cuatro caracteres en curso de edición para facilitar el trabajo.

En la matriz de edición un «*» significa un punto en la pantalla y un «.» significa un punto vacío. El trabajo del editor consiste en modificar estas configuraciones — que corresponden a las standard del VIC — y obtener las necesarias para un juego, etc... Para ello el punto que cambia de color de forma intermitente nos indica la posición del «cursor» que nos va a facilitar el trabajo de edición. Para moverlo debemos utilizar las teclas siguientes:

W: Para moverse hacia arriba

X: Para moverse hacia abajo

A: Para moverse a la izquierda

D: Para moverse a la derecha

Por lo que se refiere a las teclas de edición propiamente dichas, son las conocidas F1 a F8 y funcionan de la siguiente manera:

F1: Poner un punto en la posición.

F2: Ver los caracteres normales.

F3: Quitar un punto.

F4: Generar las instrucciones DATA para el bloque actual de caracteres.

F5: Seleccionar otro bloque de caracteres a editar. Como se observará, los caracteres se editan de cuatro en cuatro. Para escoger el bloque de cuatro caracteres que se quiere editar se pulsa F5 y, cuando aparezca el interrogante, se responde con el primer carácter del nuevo bloque.

F6: Elimina el editor y deja solamente las instrucciones DATA correspondientes a los caracteres editados.

F7: Pone en blanco el bloque de caracteres que se están editando.

Como se ha podido observar, la «salida» del editor es una serie de instrucciones DATA que se pueden almacenar en cinta o disco como un programa normal al que posteriormente se le añadirá el programa propiamente dicho que maneja estos datos. Veamos un ejemplo:

Entramos el programa y modificamos un bloque de caracteres, por ejemplo, el @-A-P-Q. Una vez realizada la modificación, pulsamos F4 (crear las instrucciones DATA) y seguidamente F6 (salir del editor conservando las instrucciones DATA). En este momento aparecerá el mensaje READY. Si hacemos LIST, obtendremos un pequeño programa con números de línea 100, 9000... y 9999. Este programa es el que podemos guardar en disco o cinta y que deberá ser el que incorporemos en el programa que necesite caracteres especiales. Como prueba haga RUN y luego POKE36869,255. En este momento, los caracteres definidos de nuevo aparecerán con el nuevo aspecto. Para volver a las condiciones normales haga POKE36869,240. En este punto (o antes de realizar la prueba anterior) se puede hacer un SAVE de este programa para su posterior utilización. En el programa que utilice este nuevo juego de caracteres se tienen que ejecutar las líneas que nos ha suministrado el editor y, en el momento de necesitar los nuevos caracteres, debe hacerse POKE36869,255, en el caso de trabajar con 8K o menos, o POKE36869,207 con más de 8K. En ambos casos, se puede volver a la normalidad con POKE36869,240 ó POKE36869,192 respectivamente.

El programa que se adjunta funciona con 8K ó menos de memoria.

(continuará)

APLICACIONES EDUCATIVAS

repasando las tablas

por **JOAN V. BAZ**



Ésta es realmente una colaboración y no una nueva sección, aunque si la cosa funcionara y la imaginación —que sabemos inagotable— de nuestros lectores se orientara hacia la colaboración...

Joan Baz es maestro en un pueblo de Tarragona; es joven y, sin embargo, ha visto mundo y será por esto que no tiene pelos en la lengua.

Nota de la Redacción: *Ponemos al VIC como testigo de que CLUB COMMODORE no ha tenido nada que ver en la redacción de este artículo. No obstante, no resistimos la tentación de poner de manifiesto una de nuestras adhesiones más inquebrantables a las tesis de Joan sobre el tema de las COLABORACIONES.*

Soy, como tantos de los que leéis esto, un «Vic-adicto». No desde hace mucho, claro, pero hay que reconocer que crea una cierta dependencia.

La razón que me impulsa a escribir estas líneas es que creo que estamos en un cierto «impasse». Me explicaré. Esta Revista debe servir para que nos pasemos información, sugerencias, ideas, entre usuarios del Vic (entre otros) y no es así. Hay pocas colaboraciones. ¿Qué es lo que pasa?

Creo que se debe al poco tiempo que hace que tanto el Vic como la Revista están en la calle, por un lado. Por otra parte, conozco gente que podrían enviar programas y no lo hace por miedo al ridículo. A mi entender eso es lo realmente ridículo. Todos somos novatos en «Vicsic», dado que es una máquina muy reciente, así que rompamos con el complejo y si queremos que la cosa funcione pongámonos a escribir. Todos ganaremos con ello. Creo que un programa regular es mejor que ninguno y puede dar lugar a comentarios y enmiendas con lo que nos pueden ayudar.

Y como el movimiento se demuestra andando, aquí tenéis mi primer programa: «Tablas». Lo escribí al poco tiempo de recibir mi Vic para ayudar a los niños con quienes trabajo a repasar las tablas de multiplicar. Después de meses de funcionamiento han quedado demostradas tres cosas.

a) Los niños disfrutan estudiando con ordenador incluso materias aburridas y repetitivas como ésta.

b) El Vic es capaz de aguantar meses de trato duro.

c) Dado que los niños han mejorado sensiblemente su conocimiento de las tablas, hemos de reconocer que el método funciona.

Antes de pasar a comentar el programa quisiera dejar bien claro que no es mi intención hacer un panegírico del Vic. En próximas colaboraciones le daré un «vistazo» crítico.

Dejando aparte los REM (remarks, comentarios que no afectan al programa en sí), empezamos en la línea:

60: Asigna nombre a las variables de los registros de sonido y pantalla.

70: Limpia el buffer *, da entrada a la variable A (tabla a repasar).

90-100: Imprime la pregunta a calcular, limpia el buffer, da entrada a la variable C (respuesta del niño).

110: Caso error (respuesta incorrecta) continúa en 170.

120-140: Respuesta correcta, imprime la pregunta y la solución, un mensaje anima al niño, en la subrutina 270 el color y el sonido realzan el mensaje.

150: Retraso de tiempo (variable T) para darle tiempo a leer.

160: Empieza de nuevo.

170: La variable N controla el número de veces que se ha equivocado en la misma pregunta; si es mayor o igual a tres, sigue en corrección (240).

180-200: Cambia el color y fondo de la pantalla, la limpia e imprime un mensaje por la equivocación.

210-230:Sonido para la equivocación (sirenas). Vuelve a la pregunta (90).

240: Corrección. Imprime la respuesta correcta.

250: Envía a 380 (sonido).

260: Empieza de nuevo.

270: Subrutina el sonido.

380-400: Sonido, empieza de nuevo.

* El buffer de teclado es una especie de depósito donde se van recogiendo las teclas que han sido apretadas antes de su ejecución; POKE 198,0 lo limpia y es buena práctica hacerlo antes de un INPUT en programas que van a ser utilizados por niños (suelen jugar con el teclado). ■

```
1 REM ****************
2 REM REPASO DE TABLAS
3 REM ****************
4 REM >>JUAN V. BAZ <<
5 REM ****************
60 S1=36876:S2=36875:S3=36878:BP=36879
70 POKE198,0:INPUT"QUE TABLA REPASAMO
S";A
80 B=INT(RND(1)*9)
90 PRINT"          "A;"X";B;"=";
100 POKE198,0:INPUTC
110 IFC<>A*BTHEN170
120 PRINT"          "A;"X";B;"=";C
130 PRINT" FELICIDADES
 HAS ACERTADO"
140 GOSUB270
150 FORT=1TO1000:NEXTT
160 RUN
170 N=N+1:IFN>=3THEN240
180 POKEBP,202:PRINT"          "A;"X";B;
"<>";C
190 PRINT"<> SIGNIFICA: NO ES   IGU
AL A..."
200 PRINT"ME TEMO QUE NO TE LO  SAB
ES MUY BIEN, INTEN-TALO DE NUEVO"
210 POKES3,15:FORL=1TO7:FORM=180TO235STE
P2:POKES1,M:FORT=1TO10:NEXTT:NEXTM
220 POKES1,0:FORM=1TO100:NEXTM:NEXTL:POK
ES3,0:POKEBP,27
230 GOTO90
240 PRINT"ESTA BIEN,TE LO DIRE:
";"       ";A;"X";B;"=";(A*B)
250 GOTO380
260 RUN
270 POKEBP,108:POKES3,15
280 FORL=148TO220STEP.7
290 POKES1,L
300 NEXTL:POKEBP,154
310 FORL=128TO200
320 POKES1,L
330 NEXTL:POKEBP,253
340 FORL=200TO128STEP-1
350 POKES1,L
360 NEXTL:POKEBP,27
370 POKES3,0:POKES1,0:RETURN
380 POKES3,15:FORL=1TO15:FORM=200TO220+L
*2:POKES1,M:NEXTM:NEXTL
390 POKES3,0:POKES1,0:FORT=1TO500:NEXT
400 RUN

READY.
```

NOTICIAS

programas para el VIC-20

En este número iniciamos una reseña de los programas para el VIC-20 que comercializa MICROELECTRÓNICA Y CONTROL, S. A. Aquí están los primeros:

PROGRAMAS EN DISCO

Ref. D-1002
Programa QLS

Registro de contactos para radioaficionados. P.V.P. **3.000.**

PROGRAMAS EN CINTA

Ref. C-128
Programa Programación lineal

Método simplex. Cálculo del valor de las variables que, satisfaciendo las restricciones, hacen máxima o mínima una función. En castellano. P.V.P. **500.**

Ref. C-129
Programa Matrices

Suma, resta, multiplicación, multiplicación por un escalar e inversión de matrices. En castellano. P.V.P. **500.**

Ref. C-131
Programa Regresiones I

Contenido:
Regresión lineal: Cálculo por mínimos cuadrados de la recta que se ajusta mejor a una nube de puntos.
Regresión múltiple: Variable dependiente en función de **n** variables independientes de grado 1. en Castellano. P.V.P. **500.**

Ref. C-132
Programa Regresiones II

Contenido:
Regresión de orden **n:** Variable dependiente en función de variable independiente de grado **n.**
Regresión exponencial: Ajuste nube de puntos a una curva exponencial.
Regresión geométrica: Ajuste de una nube de puntos a una curva geométrica. En castellano. P.V.P. **500.**

Ref. C-133
Programa Estadística I

Distribución normal, Poisson, binomial, chi-cuadrado, Student, F de Snedecor. En castellano. P.V.P. **500.**

Ref. C-134
Programa Estadística II

Contenido: Cálculo de la media, varianza y desviación tipo, tanto de la muestra como de la población, estando los datos agrupados o no.
Test de chi-cuadrado y test de Student. En castellano. P.V.P. **500.**

Ref. C-135
Programa Sistemas

Contenido:
Resolución de sistemas de **n** ecuaciones con **n** incógnitas.
Resolución de ecuaciones de grado 2 dando las soluciones tanto reales como complejas.
Cálculo de permutaciones y de combinaciones. En castellano. P.V.P. **500.**

PROGRAMAS EN CARTUCHO

Ref. C-404
Programa Sargon II Chess

Para jugar al ajedrez «contra» el VIC-20. Con 7 niveles de dificultad. Uno de los juegos más fascinantes y educativos que existen en una de las mejores versiones. P.V.P. **4.500.**

Ref. C-405
Programa Dadar Ratrace

Un juego de persecución de gran efecto en la pantalla de su televisor. P.V.P. **4.500.**

CLUBS DE USUARIOS

nunca es demasiado tarde

¡Al fin empieza esto a moverse! De Barcelona nos llega el siguiente anuncio con ruego de publicación:

SE ANUNCIA LA INMINENTE CREACIÓN DE UN CLUB DE USUARIOS DE VIC-20 EN BARCELONA.
LA PRIMERA REUNIÓN TENDRÁ LUGAR EL DÍA 13 DE ENERO DE 1983 EN: «SISTEMA», BALMES 434, TELÉFONO 211 54 40. A LAS 7 DE LA TARDE.

¡A ver si cunde el ejemplo! ¡Hasta pronto!

VIC-20

algunos trucos para el VIC-20



En la figura 1 se da el listado de algunos segmentos de programa que pueden ser de utilidad para los usuarios del VIC-20. Por supuesto, este listado no constituye un programa entero en sí mismo y deben utilizarse las partes que sean necesarias. Veamos estas partes una por una:

```
1 REM TRUCO NO 1; 255 COLORES
10 FORX=0TO255
20 POKE36879,X
30 PRINTCHR$(147)
40 FORT=1TO700:NEXTT
50 NEXTX
60 REM TRUCO NO 2; UN MENSAJE RUEDA POR
LA PANTALLA
70 POKE36867,12:REM PUEDE ESCOJER EL NUM
ERO DE LINEAS DE SU MENSAJE
80 PRINTCHR$(147)
90 PRINT"□SU MENSAJE..."
100 FORX=0TO120
110 POKE36865,X
120 NEXT
130 GOTO90
140 REM TRUCO NO 3; ROTACION DE LA PANTA
LLA
150 PRINTCHR$(147)"MAREO"
160 FORL=0TO6.28STEP.1
170 POKE36864,13+4*SIN(L)
180 POKE36865,27+4*COS(L)
190 NEXT
200 GOTO160
210 REM TRUCO NO 4; DIBUJAR LA FUNCION S
ENO
220 PRINT"■□"
230 POKE36867,22
240 POKE36865,60
250 F(8)=0:F(0)=128:F(1)=64:F(2)=32:F(3)
=16
260 F(4)=8:F(5)=4:F(6)=2:F(7)=1
270 FORQ=0TO255
280 POKE7680+Q,Q
290 POKE38400+Q,2
300 NEXTQ
310 FORQ=5120TO5120+255*8
320 POKEQ,0
330 NEXTQ
340 POKE36869,253
350 POKE36866,PEEK(36866)OR128
360 POKE36867,150
370 REM
380 REM FUNCION A DIBUJAR EN LINEA 410
390 REM
400 FORC=0TO175
410 L=45+40*SIN(C/10)
420 REM
430 REM RUTINA DE DIBUJO EN ALTA RESOLUC
ION
440 REM
450 A=5120
460 LR=L/8
470 LA=INT(LR)
480 A=A+(LA*176)
490 LR=(LR-LA)*8
500 CR=C/8
510 CA=INT(CR)
520 A=A+(CA*8)
530 A=A+LR
540 CR=INT((CR-CA)*8)
550 POKEA,PEEK(A)ORF(CR)
560 NEXTC
570 REM
580 REM ESPERAR UNA TECLA
590 REM ENTONCES VOLVER A NORMAL
600 REM
610 GETA$:IFA$=""THEN610
620 POKE36869,240
630 POKE36866,150
640 POKE36867,174
650 POKE36865,38
660 PRINT"□"

READY.
```

(Fig. 1)

TRUCO NÚM. 1 - LÍNEAS 10-50

Este pequeño programa que exhibe sucesivamente las 255 combinaciones de colores de pantalla y marco es útil para determinar, paso a paso, el juego de colores que nos puede interesar para una aplicación determinada. El POKE de la línea 20 es la clave del asunto: ejecute el programa y cuando vea la combinación deseada pulse RUN/STOP—RESTORE (para detener el programa) y seguidamente pulse PRINT X y tendrá el último valor que se ha introducido en el registro correspondiente del controlador de video. Éste es un sistema algo retorcido de resolver el problema. Lo más elegante es consultar las páginas 37 ó 134 del MANUAL DEL USUARIO.

TRUCO NÚM. 2 - LÍNEAS 70-130

Se pueden cambiar las dimensiones de la «ventana» de video del VIC utilizando algunos POKES especiales que ya se han expuesto en otras partes de nuestra Revista. Esta sección de programa usa algunos de ellos para exhibir un mensaje que se desliza por la pantalla de una manera que llama mucho la atención. En la línea 90 puede insertarse el mensaje.

TRUCO NÚM. 3 - LÍNEAS 150-200

Si es necesario marear a alguien por medios sofisticados, se puede utilizar este truco que también se beneficia de la posibilidad de jugar con los parámetros de video del VIC. Con esta rutina se consigue hacer dar vueltas al espacio dedicado a caracteres.

TRUCO NÚM. 4 - LÍNEAS 220-660

Éste es el más espectacular de todos, pues permite realizar dibujos en alta resolución sin necesidad de cartucho alguno. En el ejemplo que adjuntamos se exhibe el juego de caracteres y luego se dibuja la función seno. Como ya hemos explicado mucho del vic (VIDEO INTERFACE CHIP), dejamos a la paciencia del lector el desarrollar otras posibilidades de trazado gráfico. ■

NOTAS

ayuda para contables

Dar una lista de números por pantalla —o impresora— puede producir cierta frustración y desconsuelo si se utiliza para ello la instrucción TAB. Veamos un ejemplo:

Si el valor de la variable es cero, este número se escribirá en la columna de la izquierda y esto mismo sucederá con todos los valores de una sola cifra. Con dos o más cifras, las cantidades se alinearán por la izquierda (se suele hablar de justificación del margen izquierdo) que es lo contrario a los más elementales criterios de legibilidad y elegancia. Es evidente que se debe usar el número de dígitos (incluyendo el punto decimal) para controlar la expresión TAB. Un número tiene a la izquierda un espacio reservado para el signo y a la derecha uno para el cursor que deben tenerse en cuenta. La función LEN «cuenta» el número de caracteres de una cadena alfanumérica pero, para que nos sirva, debemos convertir la cantidad previamente en una cadena. Esto se hace con STR$. Veamos:

X = LEN (STR$ (A)) — 1

donde A es la variable numérica, debe darnos lo que buscamos. El espacio del cursor a la derecha no se tiene en cuenta por STR$ y se resta 1 para compensar el espacio de la izquierda. Si hacemos PRINT TAB (10—X) la cosa tiene aspecto de funcionar ¿no? No, aún faltan algunos detalles.

Hay una serie de posibilidades que se deben tener en cuenta. A veces se desea incluir ceros no significativos en las cantidades que se imprimen (ej.: «0038» en lugar de «38»). Con números que incluyen decimales podemos necesitar un número fijo de éstos, seguidos por los ceros no significativos que hagan falta. Todo esto afectará el posicionamiento de las cantidades.

Incuestionablemente la manera más fácil de manejar el asunto de formatear cantidades (otra palabra de argot informático, espero que quede bien claro su significado por el contexto) consiste en convertirlas en cadenas alfanuméricas:

A$ = MID$ (STR$ (A) , 2)

STR$ convierte «A» en una cadena y MID$ se utiliza para tomar el segundo carácter — es decir prescindiendo del primer espacio —. Para añadir ceros no significativos usaremos la función RIGHT$:

A$ = RIGHT$ («000000» + A$, 4)

Si «A» es un entero, esto nos dará una cadena de cuatro caracteres. El número de ceros no significativos dependerá de la longitud de «A» (ej.: «16» se verá como «0016», sin embargo «1024» quedará igual). Para abreviar todo lo anterior se puede obtener con:

A$ = RIGHT$ («000000» + MID$ (STR$ (A) , 2) , 4)

El redondeo de números tiene su pequeño truco. Lo primero es decidir qué número de decimales queremos. El siguiente es un ejemplo para dos dígitos decimales:

A = 1035.55534
A = INT (A * 100) / 100
PRINT A
1035.55

Este ejemplo «mueve» el número dos cifras a la izquierda, corta la parte decimal que queda y lo «mueve» a la derecha dos lugares otra vez. No obstante, esto no es propiamente «redondear» sino más bien «truncar» que no es lo mismo. Para redondear tenemos que tener en cuenta la importancia de los decimales que despreciamos. Así nuestro ejemplo se convierte en:

A = 1035.55534
A = INT (A * 100 + .5) / 100
PRINT A
1035.56

¡Esto sí es un redondeo! (fíjese en la segunda cifra decimal). De una forma general la segunda línea quedaría:

A = INT (A * 10∧D + .5) / 10∧D

Donde «D» es el número de decimáles deseado.

Los valores que terminan con un solo decimal o menos deben ser redondeados con ceros no significativos. De nuevo esto puede hacerse con manipulación de cadenas de caracteres:

A = 1035.59534
V = A
A = INT (ABS (A) * 100 + .5) / 100)
PRINT A
1035.6
S$ = CHR$ (32 — (V<0) * 13)
A$ = MID$ (STR$ (INT (A)), 2)
DP = INT ((A — INT (A)) * 100 + .5)
A$ = A$ + »,» + RIGTH$ («00» + MID$ (STR$ (DP), 2), 2)
A$ = RIGTH$ («[10 esp.]» + S$ + A$, 10)
PRINT A$
1035.60

Esto — quizás — tiene un aspecto algo confuso, pero lo hace todo: añade ceros no significativos, redondea, etc... Debe hacerse notar que hemos utilizado el valor ABSoluto de A antes de entrar la rutina (aunque V conserva el valor original de A). S$ será o un espacio o un signo menos dependiendo del valor de A. DP sirve para tomar los decimales de A y redondear esta cantidad. Lo siguiente es montar estas partes en un todo que será la cadena A$, — nótese que se sustituye el punto decimal utilizado en los países anglosajones por la coma que se usa aquí — y, por último, añadimos espacios en blanco y el signo con S$. En la figura 1 se dan las rutinas en forma de programa, dando como resultado la impresión justificada a la derecha con y sin ceros. ■

```
10 REM MINI NOTAS; IMPRESION JUSTIFICADA
20 REM RUTINA DE IMPRESION SIN CEROS
30 INPUTA
40 V=A
50 A=INT(ABS(A)*100+.5)/100
60 S$=CHR$(32-(V<0)*13)
70 A$=MID$(STR$(INT(A)),2)
80 DP=INT((A-INT(A))*100+.5)
90 A$=A$+","+RIGHT$("00"+MID$(STR$(DP),2
),2)
100 A$=RIGHT$("                    "+S$+A$,10)
110 PRINTTAB(12)A$
120 REM RUTINA DE IMPRESION CON CEROS
130 INPUTA
140 V=A
150 A=INT(ABS(A)*100+.5)/100
160 S$=CHR$(32-(V<0)*13)
170 A$=MID$(STR$(INT(A)),2)
180 DP=INT((A-INT(A))*100+.5)
190 A$=A$+","+RIGHT$("00"+MID$(STR$(DP),
2),2)
200 IFS$=" "THENS$="0"
210 A$=RIGHT$("0000000000"+A$,10)
220 PRINTTAB(11)S$+A$

READY.
```

(Fig. 1)

CORREO ABIERTO

Los lectores que se sientan impulsados a plantear una consulta que pueda aclarar sus dudas y —¿por qué no?— la de otros muchos, pueden manifestarla por escrito remitiéndola a: CORREO ABIERTO - «Club Commodore» - Calle Taquígrafo Serra, n.º 7, 5.ª planta - Barcelona-29. Pondremos todas nuestras luces a su disposición para las aclaraciones. ¡Que nadie se quede con sus dudas, cavilando en solitario!

Pregunta: ¿Se puede conectar una unidad de discos doble tipo 8050 ó 4040 al VIC?

Respuesta: Cuando se disponga del cartucho de interfaz IEEE-488 para el VIC (anunciado por COMMODORE) se podrá conectar al VIC cualquier periférico de la línea CBM y cualquier instrumento (este bus es muy utilizado en instrumentación) que incorpore este standard.

Pregunta: ¿Cómo puedo conectar un modem al VIC?

Respuesta: Depende de qué tipo de interfaz con el ordenador use el modem (modem significa **MO**dulador-**DEM**odulador y sirve para transmitir y recibir información a través de la línea telefónica). Si éste lleva interfaz para el bus IEEE-488 la pregunta queda resuelta en el párrafo anterior; si lleva interfaz RS 232 C, para el VIC existe un cartucho que se conecta al port de usuario y que permite enlazar con este standard serie (así como el IEEE-488 es el bus paralelo más utilizado, el RS 232 C es el más popular de los buses de datos serie). En este caso el sistema operativo (KERNAL) del VIC permite utilizarlo como si fuera un periférico más.

Pregunta: ¿Cómo se puede conectar una impresora al VIC-20?

Respuesta: A) Lo más fácil (y barato) consiste en conectar la impresora que vende COMMODORE para el VIC. No obstante, en determinadas aplicaciones, pueden necesitarse algunas características que no se incluyen en la impresora del VIC (ej., un ancho de papel mayor). En este caso:

B) Algunas impresoras llevan como enlace standard el RS 232 C (véase la pregunta anterior), en este caso se puede utilizar el cartucho de interfaz para este standard.

C) Otro de los enlaces típicos de las impresoras es el llamado TIPO CENTRONICS que no es más que un enlace paralelo de 7 ó 8 bits a nivel TTL (esto indica los niveles de tensión de las señales que se intercambian el ordenador y la impresora y que son + 5 V para el 1 lógico y 0 V para el 0). En este caso se puede conectar la impresora directamente al port de usuario del VIC que dispone de estos niveles de tensión. Entonces la información se manda en paralelo y no en serie como en el caso del RS 232 C, lo cual es más rápido pero tiene el inconveniente de no poder usar las rutinas del sistema operativo porque cada impresora tiene su sistema de intercambio de información.

Pregunta: ¿Puede invertirse hacia arriba el avance automático de la pantalla (SCROLL)?

Respuesta: Puede hacerse SCROLL invertido con una rutina en lenguaje máquina. No existe en ROM una rutina que pueda hacer esto. Así pues, hay que escribir una.

Pregunta: ¿Hay algún comando asignado a las teclas de función en el cartucho de ayuda al programador?

Respuesta: Sí, hay dos juegos de funciones que se asignan automáticamente al inicializar el cartucho (al hacer SYS 4096*7+9 ó SYS 28681). No obstante, estas funciones se pueden redefinir según las necesidades del usuario.

Pregunta: ¿Es posible devolver al VIC a una configuración de memoria sin expansión y sin quitar los cartuchos?

Respuesta: No, para trabajar como si tuviéramos el VIC con un mínimo de memoria hay que desconectar dicha memoria y para ello es **imprescindible** cerrar el interruptor de red.

Pregunta: ¿Cómo puede moverse la memoria de pantalla en el espacio de memoria del VIC?

Respuesta: Para cambiar la posición de la memoria de pantalla hay que cambiar el contenido de los cuatro bits más altos de la posición de memoria 36869. Éste es el puntero de memoria de pantalla en el controlador de video. Además, debe «decírsele» al editor de pantalla (el programa en el sistema operativo que se dedica a la gestión de la pantalla) dónde ha de hacer sus operaciones. Esto se consigue haciendo un POKE en la posición 648 con el valor del BYTE más significativo de la nueva dirección. Luego debe reinicializarse el sistema operativo pulsando simultáneamente RUN/RESTORE o con SYS 65234.

Pregunta: ¿En qué posición de memoria empieza el BASIC con los cartuchos de 8 y 16K conectados?

Respuesta: El BASIC (START OF BASIC) se localiza en 4096 decimal con más de 8K de memoria.

MARKETCLUB

- **Se busca experto en VIC-20 para colaborar en la creación y coordinación de un Club de Usuarios de VIC en Barcelona. Llamar a Srta. Rosa Romero. Teléfono 211 54 40.**
- **Vendo cartucho 16K VIC-20, por 14.000 ptas. Hago programas en Basic de Commodore (todas las versiones) bajo encargo. Desearía contactar con usuarios de Commodore en la zona de Madrid, para cambio de programas, impresiones, pokes especiales, etc... Razón: Francisco Gutiérrez. Santiago Rusiñol, 12. MADRID-3. Teléf. (91) 253 13 40. Horas comida y cena.**
- **Vendo equipo CBM 3032 y 3040 (CPU y Floppy). Interesados llamar a Miguel al (93) 300 16 27 de BARCELONA.**
- **Desearía contactar con personas interesadas en la enseñanza de la Informática a jóvenes de 9-14 años para adquirir programas y documentación. Llamar a Eduardo Guardino al (93) 209 94 49 de BARCELONA.**

MAPA DE MEMORIA BASIC 2

0000-0002	0-2	USR Jump instruction lo-hi
0003	3	General counter for Basic. Search character ':' or endline
0004	4	Scan-between-quotes flag. 00 as delimeter
0005	5	Basic input buffer pointer; # subscripts
0006	6	Default DIM flag. First character of array name
0007	7	Variable flag, type: FF=string, 00=numeric
0008	8	Integer flag, type: 80=integer, 00=floating point
0009	9	DATA scan flag; LIST quote flag; memory flag
000A	10	Subscript flag; FNx flag
000B	11	Flags for input or read, 0==input: 64=get: 152=read
000C	12	ATN sign flag: comparison evaluation flag
000D	13	input flag; suppress output if negative
000E	14	current I/O device for prompt-suppress
0011-0012	17-18	Basic integer address (for SYS, GOTO etc)
0013	19	Temporary string descriptor stack pointer
0014-0015	20-21	Last temporary string vector
0016-001E	22-30	Stack of descriptors for temporary strings
001F-0020	31-32	Pointer for number transfer
0021-0022	33-34	Misc.number pointer
0023-0027	35-39	product staging area for multiplication
0028-0029	40-41	Pointer: Start-of-Basic memory
002A-002B	42-43	Pointer: End-of-Basic, Start-of-Variables
002C-002D	44-45	Pointer:End-of-Variables,Start-of-Arrays
002E-002F	46-47	Pointer: End-of-Arrays
0030-0031	48-49	Pointer: Bottom-of-Strings (moving down)
0032-0033	50-51	Utility string pointer
0034-0035	52-53	Pointer: Limit of Basic Memory
0036-0037	54-55	Current Basic line number
0038-0039	56-57	Previous Basic line number
003A-003B	58-59	Pointer to Basic statement (for CONT)
003C-003D	60-61	Line number, current DATA line
003E-003F	62-63	Pointer to current DATA item
0040-0041	64-65	Input vector
0042-0043	66-67	Current variable name
0044-0045	68-69	Current variable address
0046-0047	70-71	Variable pointer for FOR/Next
0048-0049	72-73	Y save register-new operator save; current operator pointer
004A	74	Special mask for current operator; comparison symbol
004B-004C	75-76	Misc numeric work area; function definition pointer,lo-hi
004D-004E	77-78	Work area; pointer to string description
004F	79	Length of above string
0050	80	constant used by garbage collect routine, 3 or 7
0051-0053	81-83	Jump vector for functions
0054-0058	84-88	Misc numeric storage area
0059-005D	89-93	Misc numeric storage area
005E-0063	94-99	Accumulator#1: E,M,M,M,M,S
0064	100	Series evaluation constant pointer
0065	101	Accumulator hi-order propogation word
0066-006B	102-107	Accumulator#2
006C	108	Sign comparison, primary vs. secondary
006D	109	Low-order rounding byte for Acc#1
006E-006F	110-111	Cassette buffer length/Series Pointer
0070-0087	112-135	Subrtn: Get Basic Char; 77,78=pointer
0088-008C	136-140	RND storage and work area
008D-008F	141-143	Jiffy clock for TI and TI$
0090-0091	144-145	IRQ RAM vector,lo-hi; hardware interrupt vector
0092-0093	146-147	Break interrupt vector
0094-0095	148-149	NMI RAM interrupt vector,lo-hi
0096	150	Status word ST
0097	151	Which key depressed: 255=no key
0098	152	Shift key: 1 if depressed
0099-009A	153-154	Clock correction factor;lsb-msb; 1/30 sec increment
009B	155	Keyswitch PIA duplicate of 59410 : STOP and RVS flags
009C	156	Timing constant buffer
009D	157	Load=0, Verify=1
009E	158	# characters in keyboard buffer
009F	159	Screen reverse flag
00A0	160	IEEE-488 output flag: FF=character waiting
00A1	161	End-of-line-for-input pointer
00A3-00A4	163-164	Cursor log (row,column)
00A5	165	IEEE-488 output character buffer
00A6	166	Key image
00A7	167	0=flashing cursor, else no cursor
00A8	168	Countdown for cursor timing
00A9	169	Character under cursor
00AA	170	Cursor blink flag
00AB	171	EOT bit received
00AC	172	Input from screen/input from keyboard
00AD	173	X save flag
00AE	174	How many open files; pointer into file table
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally default of 3
00B1	177	Tape character parity
00B2	178	Byte reveived flag
00B4	180	Tape buffer character
00B5	181	Pointer in filename transfer
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Countdown for tape write; sync on tape header
00BB	187	Tape buffer#1 count
00BC	188	Tape buffer#2 count
00BD	189	Write leader count; Read pass1/pass2
00BE	190	Write new byte; Read error flag
00BF	191	Write start bit; Read bit seq error
00C0	192	Pass 1 error log pointer
00C1	193	Pass 2 error correction pointer
00C2	194	Current function; 0-Scan; 1-15=Count; $40=Load; $80=End
00C3	195	Read checksum; Write leader length
00C4-00C5	196-197	Pointer to screen line
00C6	198	Column position of cursor on above line (0-79)
00C7-00C8	199-200	Utility pointer: tape buffer,scrolling
00C9-00CA	201-202	Tape end address/end of current program
00CB-00CC	203-204	Tape timing constants
00CD	205	Flag for quote mode 0=direct cursor, else programmed cursor
00CE	206	Timer 1 enabled for tape read; 00=disabled
00CF	207	EOT signal received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current logical file number
00D3	211	Current secondary addrs, or R/W command
00D4	212	Current device number
00D5	213	Line length (40 or 80) for screen
00D6-00D7	214-215	Start of tape buffer, address
00D8	216	Line where cursor lives
00D9	217	Last key input; buffer checksum; bit buffer
00DA-00DB	218-219	Pointer to current file name
00DC	220	Number of keyboard INSERTs outstanding
00DD	221	Write shift word/Receive input character
00DE	222	#blocks remaining to write/read
00DF	223	Serial word buffer
00E0-00F8	224-248	Screen line table: hi order address & line wrap
00F9	249	Interrupt driver flag for cassette#1 status switch
00FA	250	Interrupt driver flag for cassette#2 status switch
00FB-00FC	251-252	Tape start address
0100-010A	256-266	Binary to ASII conversion area
0100-013E	256-318	Tape read error log for correction
0100-01FF	256-511	Processor stack area
0200-0250	512-592	Basic input buffer
0200-0201	512-513	Program counter
0202	514	is processor status
0203	515	is accumulator
0204	516	X index
0205	517	Y index
0206	518	stack pointer
0207-0208	519-520	user modifiable IRQ
0251-025A	593-602	Logical file number table
025B-0264	603-612	Device number table
0265-026E	613-622	Secondary address, or R/W cmd, table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 buffer
033A-03F9	826-1017	Tape#2 buffer
03FA-03FB	1018-1019	Vector for Machine Language Monitor
0400-7FFF	1024-32767	Available RAM including expansion
8000-8FFF	32768-36863	Video RAM
9000-BFFF	36864-49151	Available ROM expansion area
C000-E0F8	49152-57592	Microsoft Basic interpreter
E0F9-E7FF	57593-59391	Keyboard, screen, interrupt programs
E810-E813	59408-59411	PIA 1 - Keyboard I/O
E820-E823	59424-59427	PIA 2 - IEEE-488 I/O
E840-E84F	59456-59471	VIA - I/O and timers
F000-FFFF	61440-65535	Reset, tape, diagnostics, monitor

VIC-20

EL ORDENADOR PERSONAL AMPLIABLE CON COLOR Y SONIDO.



49.500 Ptas.
COLOR-SONIDO

Así es el VIC-20

- Lenguaje BASIC extendido.
- Sistema operativo COMMODORE.
- 5 K RAM ampliable a 32 K.
- 16 colores, 4 generadores de sonido.
- 66 caracteres gráficos.
- Periféricos disponibles:
  - Cassette.
  - Impresora de agujas.
  - Unidad de disco de 170 K.

Así hace las cosas el VIC-20

- Enseña informática.
- Efectúa todo tipo de cálculos matemáticos.
- Realiza funciones docentes.
- Se encarga de múltiples tareas profesionales.
- Proporciona divertidos momentos de ocio.
- Ayuda a planificar labores domésticas.
- Hace todas las aplicaciones que Vd. imagine.





GRATIS

Con la adquisición de su VIC-20 recibirá además:
- MANUAL DEL USUARIO.
- INTRODUCCION AL LENGUAJE DE PROGRAMACION BASIC.
- Y 17 PROGRAMAS DE PRACTICAS (en dos cassettes).



commodore COMPUTER

Distribuidor exclusivo para España:

Microelectrónica y Control, S.A.
Taquígrafo Serra, 7 5.°. Barcelona-29
Princesa, 47 3.° G. Madrid-8

De venta en tiendas especializadas.

SLOGAN